Lab 2 - CSPs – Artificial Intelligence

Lab group 5

1. Modeling the problem as CSP:

Variables: All colors, all pets, all smokes, all drinks , all nationalities. 5 of each.
Domains: The number of each house, from 1 to 5.
Constraints: No house number can have two of any variable. And no house can have the same variable as any other house, plus the long list of constraints given in the problem.

2. How big is the state space?

To find the size of the state space, the size of the domain and the number of variables must be known. The size of the Domain for each variable is 6 because houses can also have no number. Each house has 25 variables, which is 5 types of variables(color, nation, drink, smoke, pet) and each type contains 5 variables, so the state space is 6^{25}.

4. Results

MCV: most constrained variable
LCV: least constraining value
FC: Forward checking

Methood: AC3 + MCV + LCV
Results: 0.011s

Methood: AC3 + MCV
Results: 0.002s

Methood: AC3
Results: 0.002s

Methood: FC + MCV + LCV
Results: 5.059s

Methood: FC + MCV
Results: 1.586s

Methood: FC
Results: 0.001s

To begin with we are selecting between using AC3 and forward checking. We should expect that AC3 should work better due to the intelligent design of throwing out impossible elements from domains of not-yet-assigned variables. And this is exactly what we see.

In theory we should have faster results when using both MCV and LCV. But this is not the case, neither for AC3 nor FC. We could say that this is only due to change, highly unlikely though since this trend is seen in every run of our program. The reason possibly has to do with the structure of this particular problem. Even though LCV should be more guaranteed to give you a solution in each run (one run being a single assignment of the first variable, one branch from the first node), the branching factor of subsequent variables is higher due to the fact that we are selecting the value that removes the smallest number of values from the domains of those variables which are not in the current assignment. So the search tree is bigger for each run.

It's surprising that the fastest method is using FC only. In that case, each variable and value is equally likely to be chosen, which obviously contributes to a faster solution in this case, compared to heuristics like MCV and LCV.